Skeleton in the Euclidean closet

Zalán Gyenis*

August 21, 2017

Dedicated to the 60th birthday of András Kornai and László Kálmán

Abstract

Euclidean Automata have been introduced in Kornai [Kor14a] to model a phenomenon known as "being in conflicted states". This brief note gives a further look on Euclidean Automata and takes the first steps in studying skeleta and representability and the logical characterization of languages accepted by Euclidean Automata.

Keywords: Euclidean Automata, Skeleta.

1 Introduction

Euclidean Automata (EA) has been introduced, motivated and further studied in [Kor14a] and [Kor14b]. EA are slight generalizations of the classical finite state automata: EA can take continuous parameters as input and are used in [Kor14a] to analyze the situation of being in a conflicted state. Intuitively, being in a conflicted state is modeled by an EA not as a single state (of the EA) but rather as a set of "nondeterministic" states that are represented as overlapping parts of the input parameter space. Let me recall the precise definition of Euclidean Automata.

Definition 1.1 ([Kor14b]). A Euclidean automaton (EA) over a parameter space Σ is defined as a 4-tuple (Q, I, F, α) where $Q \subseteq 2^{\Sigma}$ is a finite set of states given as subsets of Σ; $I \subseteq Q$ is the set of initial states; $F \subseteq Q$ is the set of accepting states; and $\alpha : \Sigma \times Q \to Q$ is the transition function that assigns for each parameter setting $v \in \Sigma$ and each state $q \in Q$ a next state $\alpha(v, q)$ that satisfies $v \in \alpha(v, q)$. □

In [Kor14a] the EA is called deterministic if $q \cap s = \emptyset$ for different $q, s \in Q$ and complete if $\cup_{q \in Q} q = \Sigma$. Throughout we will work with complete EA's only, the reason is that for $v \in \Sigma - \cup_{q \in Q} q$ the condition $v \in \alpha(v, q)$ does not make sense, hence

*Department of Logic, Jagiellonian University and Department of Logic, Eötvös University

either one keeps α to be undefined on certain input parameters v or switches to an equivalent EA with parameter space $\cup_{q\in Q} q$. For simplicity we assume throughout that the set of initial states I contains a unique state which we denote by `start`. If one permits several initial states he needs to complicate the results accordingly. In applications drawn in [Kor14a, Kor14b] the alphabet Σ consists of vectors from a continuous parameter space, typically $\mathbb{R}^n$, however it also makes sense to consider the definition of an EA when Σ is a finite set, especially if one considers skeleta of EA's, as we do in Section 2.

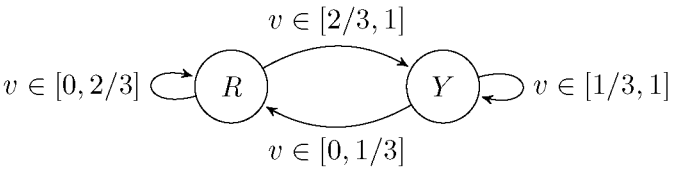
A typical application in [Kor14a] is the heap (Sorites paradox) presented in the Sainsbury and Williamson [SW95] manner: Consider the line segment $[0, 1]$ colored so that the left-hand region is red, and there is a very fine, continuous, gradual change of shades reaching the right-hand side region that is colored yellow. The line is covered by a tiny window that exposes only a small region. We move the window very slowly starting from the left-hand side towards right and after each move one is asked about the color of the segment exposed by the window. But the window is so small relative to the line segment that in no position can you tell the difference in color between what you can see at the two sides of the window. It seems that you must call every region red after every move, and thus you find yourself in the paradoxical situation calling a yellow region red.

2 Skeleta and representability

Kornai modeled the heap paradox by an EA in a similar manner as we do below (to make life easier we give a somewhat simplified model, but the differences are inessential for the sake of the example). Let's say $[0, \frac{1}{3}]$ is 'clearly red', $[\frac{2}{3}, 1]$ is 'clearly yellow' and $[\frac{1}{3}, \frac{2}{3}]$ is this 'hard to tell, orange' range. Our EA will have 2 states: red (R) and yellow (Y) respectively with $R = [0, \frac{2}{3}]$ and $Y = [\frac{1}{3}, 1]$. Note that the two states overlap exactly in the 'problematic' region. Starting from the red state the machine gets input from the continuous parameter space $\Sigma = [0, 1]$. The machine is defined as follows:

$$\alpha(v, R) = \begin{cases} R & \text{if } v \in [0, \frac{2}{3}] \\ Y & \text{otherwise.} \end{cases}, \qquad \alpha(v, Y) = \begin{cases} Y & \text{if } v \in [\frac{1}{3}, 1] \\ R & \text{otherwise.} \end{cases}$$

In figure:



In the entire fuzzy orangish region $[\frac{1}{3}, \frac{2}{3}]$ the model shows hysteresis: if it came from the red side it will output red, if it came from the yellow side it will output yellow.

To get a better understanding of how EA works, Kornai hints at skeletonizing EA's. The skeleton of an EA is defined in [Kor14b] as follows.

Definition 2.1 ([Kor14b])**.** The skeleton of an EA is a standard FSA whose alphabet corresponds to canonical representatives from each Boolean atom of Q. □

In the deterministic case (where all the states of the EA are disjoint) there is a correspondence between input letters and automaton states. However, in the nondeterministic case (where states are not necessarily disjoint) we may not be able to select distinct canonical representatives for each state (or for the Boolean atoms). In this case skeleta should be understand as a "subjective EA" (cf. [Kor14b]). The definition seems a bit vague as it is not completely clear how to chose the so called canonical representatives (or the subjective representatives), moreover, Q may have no Boolean atoms (cf. Example 2.4 below). A key for the clarification is the observation that some inputs are totally indistinguishable no matter what state the machine is in. To obtain a definition for the general case, fix an EA $\alpha : \Sigma \times Q \to Q$ and for $v, w \in \Sigma$ write

$$v \sim w \quad \Longleftrightarrow \quad (\forall q \in Q)\ \alpha(v, q) = \alpha(w, q) \tag{1}$$

Then $\sim$ is an equivalence relation on Σ. Moreover it is a congruence of α as for any input sequences $\langle v_1, \ldots, v_n \rangle$ and $\langle w_1, \ldots, w_n \rangle$, $v_i \sim w_i$ implies

$$\begin{aligned} \alpha(v_1, \ldots, v_n, \texttt{start}) &= \alpha(v_n, \alpha(v_{n-1}(\cdots, \alpha(v_1, \texttt{start})))) && (2)\\ &= \alpha(v_n, \alpha(v_{n-1}(\cdots, \alpha(w_1, \texttt{start})))) && (3)\\ &= \cdots && (4)\\ &= \alpha(w_1, \ldots, w_n, \texttt{start}) && (5) \end{aligned}$$

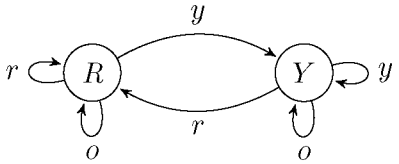
After this preparation we can redefine the concept of skeleta as follows.

Definition 2.2. The skeleton of the EA $\alpha : \Sigma \times Q \to \Sigma$ is the standard FSA $\bar{\alpha} : \Sigma/_\sim \times Q \to Q$ defined by the equation

$$\bar{\alpha}(v/_\sim, q) = \alpha(v, q)$$

Since $\sim$ is a congruence, $\bar{\alpha}$ is well-defined. □

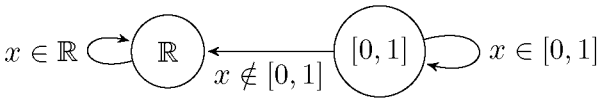
If we apply Definition 2.2 to the heap example given above we end up in the finite state machine figured below, which, unsurprisingly, is exactly the FSA sketched in [Kor14a]. (Here the input letters r, y and o stand for red, yellow and orange, respectively)



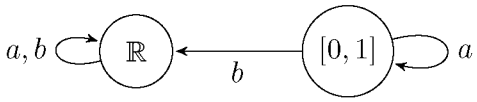
Observe that $\Sigma/_\sim$ is always finite. This is because the original EA has finitely many states only, hence we can have a finite number of possibilities not to fulfill $\alpha(v,q) = \alpha(w,q)$ for input letters v, w. This results in a finite number of equivalence classes of $\sim$. Unfortunately, $\bar{\alpha}$ is no longer an EA as $Q \not\subseteq 2^{\Sigma/_\sim}$. It would be handy to define the skeleton of an EA as another EA over the finite alphabet $\Sigma/_\sim$ by letting $Q/_\sim = \{q/_\sim : q \in Q\}$ where $q/_\sim = \{v/_\sim : v \in q\}$. However, the automaton $\beta : \Sigma/_\sim \times Q/_\sim \to Q/_\sim$ defined in the obvious manner $\beta(v/_\sim, q/_\sim) = \alpha(v,q)/_\sim$ is not always well defined as the next examples show.

Example 2.3. Below we give an example for an EA the skeleton of which can be represented as an EA. Let the alphabet (parameter space) be $\Sigma = \mathbb{R}$ and the set of states is $Q = \{\mathbb{R}, [0,1]\}$. Let α be the EA figured below defined by the equations

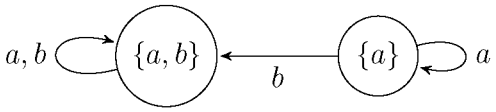
$$\alpha(x, \mathbb{R}) = \mathbb{R}, \qquad \alpha(x, [0,1]) = \begin{cases} [0,1] & \text{if } x \in [0,1] \\ \mathbb{R} & \text{otherwise.} \end{cases}$$



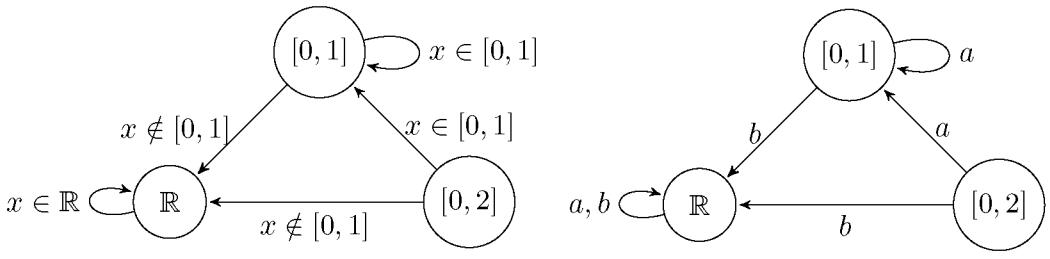
The equivalence relation $\sim$ will have two classes: $\Sigma/_\sim = \{[0,1], \mathbb{R} - [0,1]\} = \{a, b\}$ and the skeleton $\bar{\alpha}$ looks like



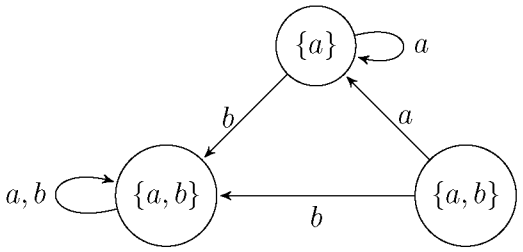
Since $\mathbb{R}/_\sim = \{a, b\}$ and $[0,1]/_\sim = \{a\}$, the skeleton is a EA over $Q/_\sim = \{a, b\}$:



Example 2.4. A small modification on Example 2.3 prevents the skeleton to be represented by an EA. Here Σ is as before but $Q = \{\mathbb{R}, [0,1], [0,2]\}$. The EA α is as figured below on the left-hand side.



The equivalence relation $\sim$ has two classes again: $\Sigma/_\sim = \{[0,1], \mathbb{R} - [0,1]\} = \{a, b\}$ (note that elements of $[0,2] - [0,1]$ behave exactly the same way elements of $\mathbb{R} - [0,1]$ do). Thus the skeleton can be figured as above on the right-hand side. Note, however, that $Q/_\sim = \{\{a, b\}, \{a\}\}$, hence the 'EA representation'



does not make sense as one cannot use the same state differently.

The previous two examples raise the question of representability. In this paper we could only give a sufficient condition, the general case definitely would require non-trivial extra work.

Definition 2.5. The EA $\alpha : \Sigma \times Q \to Q$ is said to be *localizable* if for every state $q \in Q$ there is a parameter $v \in \Sigma$ such that $v \in q - \cup_{r \in Q, r \neq q} r$ (that is, v belongs only to the state q). Localizability means that every state has an eigenparameter, a parameter which is characteristic of the state. □

In general, a state of a localizable EA can have many different eigenparameters, thus one rather speaks about the set of eigenparameters associated with a given state.

Proposition 2.6. Skeleta of localizable EA are isomorphic to EA.

Proof. The idea is that if α is localizable, then $\sim$ extends to a congruence of the state space Q. That is, if we let $Q/_\sim = \{q/_\sim : q \in Q\}$ where $q/_\sim = \{v/_\sim : v \in q\}$, then the automaton $\beta : \Sigma/_\sim \times Q/_\sim \to Q/_\sim$ defined by the equality

$$\beta(v/_\sim, q/_\sim) = \alpha(v, q)/_\sim$$

will be well-defined. As $Q/_\sim \subseteq 2^{\Sigma/_\sim}$, β will be an EA.

By localizability for every state q there is a parameter v_q, an eigenparameter of q. For two distinct states q and r the corresponding eigenparameters cannot be $\sim$-congruent, because $\alpha(v_q, q) = q$ but q does not contain v_r, hence $\alpha(v_r, q) \neq q$. The same argument show that $\alpha(v_q, s) = q$ for every state $s \in Q$. Therefore all eigenparameters associated with a given state are equivalent, and each state can be identified with that equivalence class, that is, there is a bijection between Q and $Q/_\sim$. It follows that $\alpha(v/_\sim, q/_\sim) = \alpha(v/_\sim, q)$. Finally, $\sim$ is defined in such a manner that $\alpha(v, q) = \alpha(w, q)$ whenever $v \sim w$. Thus $\alpha(v/_\sim, q)$ is well-defined, hence β is well-defined, too. ■

Example 2.3 shows an EA which is not localizable (the state $[0, 1]$ does not have an eigenparameter), still its skeleton can be represented by an EA. This means that localizability is not necessary for being representable by an EA.

Representation of standard finite state automata can be understood (at least) in two different ways.

Definition 2.7. Let Ω be a finite alphabet and R a set of states. The FSA $\delta : \Omega \times R \to R$ is representable by an EA if there is EA α over Ω such that α and δ are isomorphic.

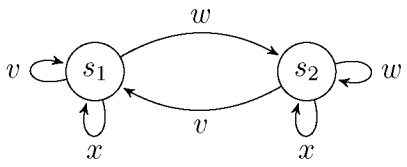
We say that α is representable in the general sense by an EA if there is a parameter space $\Sigma \supset \Omega$ and an EA α over Σ such that $\alpha \restriction \Omega$ is isomorphic to δ. □

For an FSA $\delta : \Omega \times R \to R$ and a state $s \in R$ let us denote by $[s]_{\text{in}}$ the set $\{v \in \Omega : (\exists p \in R)\delta(v, p) = s\}$.

Proposition 2.8. Let $\delta : \Omega \times R \to R$ be an FSA with the property that there are no distinct states $s_1, s_2 \in R$ such that $[s_1]_{\text{in}} = [s_2]_{\text{in}}$. Then δ is representable by an EA.

Proof. The mapping $s \mapsto [s]_{\text{in}}$ is an isomorphism. ■

Example 2.9. Consider the following FSA over the alphabet $\{v, w, x\}$.



The sets of incoming edges $[s_1]_{\text{in}} = \{v, x\}$ and $[s_2]_{\text{in}} = \{w, x\}$ are different, thus after replacing s_i by $[s_i]_{\text{in}}$ we get an isomorphic Euclidean automaton.

Unfortunately, the condition in Proposition 2.8 is not necessary: one can construct an EA that does not satisfy that condition. Here is an easy example. Take a set S and a partition of S into non-empty sets S_1 and S_2. Take S to be the alphabet and put $Q = \{S, S_1, S_2\}$ as the set of states. The automaton is defined by $\alpha(v, q) = S$ for any $v \in S$ and $q \in Q$. Then $[S_1]_{\text{in}} = [S_2]_{\text{in}} = \emptyset$.

Connections with homomorphisms. In automata theory several different types of homomorphisms between automata are defined such as state-homomorphism, alphabet-homomorphisms, etc. Since states of Euclidean automata are subsets of the alphabet, there is a natural way to generalize these concepts: homomorphisms between Euclidean Automata was defined in [Kor14b] as follows.

Definition 2.10. A homomorphism from EA $\alpha : \Sigma \times Q \to Q$ to another EA $\beta : \Omega \times S \to S$ is a mapping $h : \Sigma \to \Omega$ such that the following stipulations hold.

- $h(\texttt{start}_\alpha) = \texttt{start}_\beta$;
- h extends to a mapping $h : Q \to S$ in the natural way;
- $h\big(\alpha(v_1, \ldots, v_n, \texttt{start})\big) = \beta(h(v_1), \ldots, h(v_n), \texttt{start})$.

□

By Proposition 2.6 skeleta of localizable EA remain Euclidean: For a localizable EA $\alpha : \Sigma \times Q \to Q$ the congruence $\sim$ defined by (1) extends to a congruence of the state space Q. That is, if we let $Q/_\sim = \{q/_\sim : q \in Q\}$ where $q/_\sim = \{v/_\sim : v \in q\}$, then the automaton $\beta : \Sigma/_\sim \times Q/_\sim \to Q/_\sim$ defined by the equality

$$\beta(v/_\sim, q/_\sim) = \alpha(v, q)/_\sim$$

is an EA. Let us denote this β by $\tilde{\alpha}$. It is very easy to check (cf. the proof of Proposition 2.6), that $\tilde{\alpha}$ and the skeleton $\bar{\alpha}$ defined in 2.2 are isomorphic. Therefore we will call $\tilde{\alpha}$ also a skeleton of α.

Now, if α is localizable, then $\tilde{\alpha}$ is a homomorphic image of α. For, write $h(v) = v/_\sim$, where $\sim$ is the congruence defined by (1). The first two items of Definition 2.10 follows from the proof of Proposition 2.6 and the third item is the very definition of $\tilde{\alpha}$ as $h\big(\alpha(v, q)\big) = \alpha(v, q)/_\sim = \tilde{\alpha}(v/_\sim, q/_\sim)$.

An important consequence is that localizable EA's are categorical objects in the sense that the class of all such automata is closed under the homomorphism introduced in Definition 2.10, and skeleta form a closed subcategory of the category of all localizable EA.

3 Languages accepted by EA

In this section we turn to a logical characterization of the languages that can be accepted by EA. Let $\delta : \Omega \times R \to R$ be a standard FSA. The language of α is the set $L_\alpha \subseteq \Omega^*$ defined as

$$L_\alpha = \{w \in \Omega^* : \ \delta(w, \mathtt{start}) = \mathtt{final}\}.$$

This definition clearly makes sense even if Ω is infinite. Therefore one can define without any difficulty when a Euclidean automata $\alpha : \Sigma \times Q \to Q$ accepts a language $L \subseteq \Sigma^*$: if and only if $L = L_\alpha$.

This definition, however, may not be satisfactory enough when Σ is infinite. The reason is that one might like to say that the skeleton of an EA accepts the same language as the original EA when restricted to the language of the skeleton. More precisely one can consider the skeleton acting on a subset of the original alphabet: pick a representative from each of the equivalence class of the alphabet of the skeleton. Then the skeleton and the original EA shows the same behavior on each input string. This motivates the next definition.

Definition 3.1. Suppose $\alpha : \Sigma \times Q \to Q$ is an EA, where Σ is allowed to be infinite. Let $\Omega \subseteq \Sigma$ be a finite subalphabet and $L \subseteq \Omega^*$ a language. Then α is said to accept L in the general sense if $L_\alpha \restriction \Omega^* = L$. □

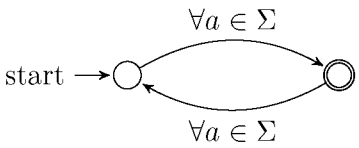
Proposition 3.2. Each regular language can be accepted in the general sense by an EA.

Proof. Let L be a regular language over Ω and $\delta : \Omega \times R \to R$ the FSA that accepts L. By adding new letters to Ω one can reach that in this expanded language (denote it by Σ) the FSA satisfies $[s]_{\text{in}} \neq [s']_{\text{in}}$ for all states $s, s' \in R$ by defining the action of the new letters carefully. The resulting FSA is representable by an EA α applying Proposition 2.8. It is clear that $L_\alpha \restriction \Omega^* = L$. The proof reveals that more is true: any language accepted by an FSA over a possibly infinite alphabet can also be accepted in the general sense by an EA. If Ω is finite then Σ can chosen to be finite; otherwise they can have the same infinite cardinality. ■

It is obvious that every language accepted by an EA is regular (because EA are special FSA), thus the question of which languages are accepted in the general sense is settled. The cheat here is that we are allowed to enlarge the alphabet. Is it true that keeping the same alphabet, for every FSA $\delta : \Sigma \times R \to R$ there is a EA $\alpha : \Sigma \times Q \to Q$ such that $L_\delta = L_\alpha$? If the alphabet is finite, then the answer obviously is 'no'. This is because over a finite alphabet Σ one can define at most finitely many EA as the set of states Q should be a subset of 2^Σ which is still finite. But what about the infinite case where one can have any finite number of states? The answer is still 'no' but for different reasons: the language that contains words having odd length (over any alphabet Σ) can be accepted by an FSA but cannot be accepted by any EA.

Example 3.3. No EA can accept the language containing words having odd length.

Proof. The language $L = \{w \in \Sigma^* : |w| \text{ is odd } \}$ is accepted by the FSA figured below.



Suppose there is an Euclidean automaton (Q, I, F, α) that accepts L. The first input letter v_1 can be any letter from Σ, hence after the first transition $\alpha(v_1, \texttt{start}) = q_1$, the resulting state should q_1 should contain v_1 by Euclideanity. This means $q_1 = \Sigma$. The second letter v_2 is also arbitrary, hence after the second transition $\alpha(v_2, q_1) = q_2$ we get similarly that $q_2 = \Sigma$. This means $q_1 = q_2$ and in fact continuing the argument one get the conclusion that there can be only one state $Q = \{\Sigma\}$. But such an EA accepts all strings and not just the ones having odd length. ■

It is known that regular languages are exactly the languages that can be defined in monadic second order logic [Bu60, El61]. Let us recall some of the basic definitions to make everything clear. Let Σ be an alphabet (possibly infinite) and let $w =$

$\langle w_1, \ldots, w_n \rangle$ be a word in Σ^*. Such a word can be represented by the relational structure

$$\mathbf{w} = \big(\{1, \ldots, n\}, <^{\mathbf{w}}, (Q_v^{\mathbf{w}})_{v \in \Sigma}\big)$$

called the word model for w, where $<^{\mathbf{w}}$ is the usual ordering on the domain of w and $Q_v^{\mathbf{w}}$ are unary predicates collecting for each letter $v \in \Sigma$ those letter positions of w which carry v:

$$Q_v^{\mathbf{w}} = \{i : \ w_i = v\}$$

The corresponding first-order language $FO(\Sigma)$ has variables $x, y, \ldots$ and built up the grammar

$$\varphi ::= \ Q_v(x) \mid x < y \mid \neg\varphi \mid \varphi \vee \psi \mid \exists x \varphi$$

The language defined by a formula φ is $L_\varphi = \{w \in \Sigma^* : \ \mathbf{w} \models \varphi\}$, where the satisfaction relation $\models$ is defined in the usual way. For example the language where "every a is immediately followed by a b" can be defined by the formula

$$\forall x \big(Q_a(x) \ \rightarrow \ \exists y (y = x + 1 \ \wedge \ Q_b(y))\big)$$

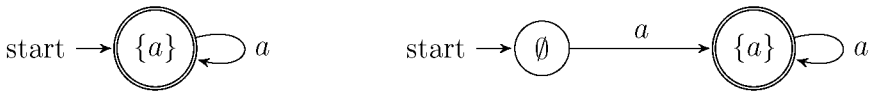
where $y = x+1$ has the usual definition $x < y \wedge \neg\exists z(x < z \wedge z < y)$. A non-example could be $L = \{a^{2^n} : \ n \in \mathbb{N}\}$ which is not expressible by a first order formula [Esp12].

Monadic second order logic $MSO(\Sigma)$ is an extension of first order logic with variables $X, Y, \ldots$ ranging over sets of elements of models. The corresponding atomic formulas $X(x)$ are also introduced with the intended meaning "x belongs to X". Clearly $MSO(\Sigma)$ is more expressive than $FO(\Sigma)$ but not vice-versa as the next theorem shows:

Theorem 3.4 (Büchi [Bu60], Elgot [El61]). A language (over a finite alphabet) is recognizable by a finite state automaton if and only if it is $MSO(\Sigma)$-definable, and both conversions, from automata to formulas and vice versa, are effective.

Thus, regular languages are exactly the monadic second order definable languages. However, examples suggests that languages accepted by Euclidean automata are first order definable:

Example 3.5. For $\Sigma = \{a\}$ we must have $Q \subseteq \{\emptyset, \{a\}\}$ and thus there are exactly two non-isomorphic EA, figured below



The languages accepted by the automata are $L_1 = \{a^n : \ n \geq 0\}$ and $L_2 = \{a^n : \ n \geq 1\}$. Both languages are definable in the language $FO(\Sigma)$, respectively by the formulas $\forall x Q_a(x)$ and $\exists x Q_a(x) \wedge \forall x Q_a(x)$.

Example 3.6. For $\Sigma = \{a, b\}$ the number of variations is larger than before as $Q \subseteq \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$ gives more possibilities. We will not draw all the non-isomorphic EA's here, but one can check easily that the languages that can be accepted by EA over Σ are of the form "all sequences of a's and $b's$ + if we wish we can prescribe the first and last letter". For example such an L can contain all words starting with an a. In any case a $FO(\Sigma)$-characterization can be easily given. (As we prove next that languages accepted by EA are first order definable, we omit the details of the rather painstaking checking of this claim).

Indeed, we prove that languages accepted by Euclidean automata are FO-definable.

Proposition 3.7. For every finite alphabet Σ there is a corresponding first order language FO such that each language accepted by an Euclidean automaton can be defined by a FO-formula.

Proof. Assume (Q, I, F, α) is a Euclidean automaton over the finite parameter space Σ. Take the first order language $FO(\Sigma)$ described above and expand this language by new unary predicates R_i for $i < 2^{|\Sigma|}$. We have to find a first order formula that expresses in any given word model $\mathbf{w}$ that α accepts w. The formula in question will state the existence of a successful run of the EA. As α has finitely many state, we can enumerate them by $Q = \{q_i : i < 2^{|\Sigma|}\}$. Each state q_i of the EA will be encoded by the predicate R_i. We need to express the followings: (1) in each turn the machine can be only in one state; (2) the starting state is q_0; (3) if we are in position x and the next position is y, then we applied one of the letters, i.e. for one of the $v \in \Sigma$ we have $Q_v(x)$, and thus the next state should be the one prescribed by $\alpha(v, q_i) = q_j$; (4) the last position is a final state. Thus α accepts w if and only if

$$\mathbf{w} \models \quad \Big(\bigwedge_{i \neq j} \forall x \neg (R_i(x) \wedge R_j(x)) \Big) \tag{6}$$

$$\wedge \quad \forall x(\mathtt{first}(x) \to R_0(x)) \tag{7}$$

$$\wedge \quad \forall x \forall y(x = y + 1 \to \bigvee_{\alpha(v,q_i)=q_j} (R_i(x) \wedge Q_v(x) \wedge R_j(y))) \tag{8}$$

$$\wedge \quad \forall x(\mathtt{last}(x) \to \bigvee_{(\exists q \in F)\alpha(v,q_i)=q} (R_i(x) \wedge Q_v(x))) \tag{9}$$

Since the empty word satisfies this sentence, if α does not accept the empty word, then a corresponding clause such as $\exists x(x = x)$ should be added. $\mathtt{first}(x)$ and $\mathtt{last}(x)$ are respectively the formulas $\neg\exists y(y < x)$ and $\neg\exists y(x < y)$. ■

If the alphabet Σ is not finite, then a similar argument shows that languages accepted by EA can be defined by first order formulas that are allowed to contain infinite disjunctions having an infinite vocabulary (i.e. we use the logic $FO_{\infty\omega}$).

Recall that *finiteness* is a property that cannot be expressed in first order logic. Indeed, by the compactness theorem if a formula holds in all finite models, then it

should also hold in an infinite model. This can be seen as one of the main reasons why languages that can be accepted by finite state automata cannot be defined in first order logic (and one needs monadic second order logic). Even if the alphabet is fixed, an FSA can have an arbitrary finite number of states and we do not have any control, in terms of first order logic, over the number of states. As we already seen, there are only finitely many EA over a finite alphabet. That is, if we fix the alphabet, then there is a fixed upper bound on the possible number of states, depending only on the size of the alphabet. This allows us to bypass the problem of non-definability of finiteness: using first order logic it is easy to define models having size at most n, for a fixed finite number n. This is the key for Proposition 3.7.

As we already mentioned, there are only finitely many EA over a finite alphabet. Therefore not every first order definable language can be accepted by an EA (there are infinitely many first order definable languages). Then what is the logic that is exactly as expressible as Euclidean automata? As the number of states is limited, the set of EA do not have any extensive closure property (such as closed under direct product, unions, etc). This suggests use the vague idea that EA are not logical in the sense of expressibility. Of course it is not clear how to define 'logicality' in a precise manner.

Acknowledgement

I am grateful to the Reviewer for his/her careful reading of the manuscript and the helpful suggestions. I wish to acknowledge the Premium Postdoctoral Grant of the Hungarian Academy of Sciences hosted by the Logic Department of the Loránd Eötvös University.

References

[Bu60] Büchi, J.R., (1960) Weak second-order arithmetic and finite automata. Z. Math. Logik Grundl. Math. **6**, pp. 66–92.

[El61] Elgot, C.C., (1961) Decision problems of finite automata design and related arithmetics. Trans. Amer. Math. Soc. **98**, pp. 21–52.

[Kor14a] Kornai, András (2014) Euclidean Automata. In: Implementing Selves with Safe Motivational Systems and Self-Improvement, 2014.03.24–2014.03.26, Los Angeles, USA.

[Kor14b] Kornai, András (2014) Finite automata with continuous input. In: S. Bensch and R. Freund and F. Otto (eds.) Short Papers from the Sixth Workshop on Non-Classical Models of Automata and Applications.

[SW95] Sainsbury, M., and Williamson, T. (1995) Sorites. In Hale, B., and Wright, C., eds., Blackwell Companion to the Philosophy of Language. Blackwell.

[Esp12] Esparza, Javier (2012) Automata theory, an algorithmic approach. Lecture notes, http://www.fi.muni.cz/usr/kretinsky/MSO_Javier_Esparza.pdf, Accessed: Aug 21, 2017.